Date: November 18, 2009

C# Sample Application
for the
CV-5000 .NET Library

Prepared by:

CONQUEST CONSULTING LLC

www.conquestsolutions.com

Copyright © 2009

This document contains information which is the property of Conquest Consulting LLC.
Except for rights expressly granted by contract, this document may not, in whole or in part, be duplicated or disclosed without prior written permission.

Revision History

Revision	Release Status	Date
1.0.0.0	Initial Release	2009-11-18

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

1 C# CV-5000 SAMPLE PROGRAM

This program will demonstrate the use of the CV-5000 .NET Library in a simple program to log measurement data and part images sent from the CV-5000. This example assumes you are familiar with C# and Visual Studio 2008.

NOTE - Before you create the sample program, you must set the IP address in the CV-5000 and configure the CV-5000 to output all measurement data and acquired images to be sent via the Ethernet.

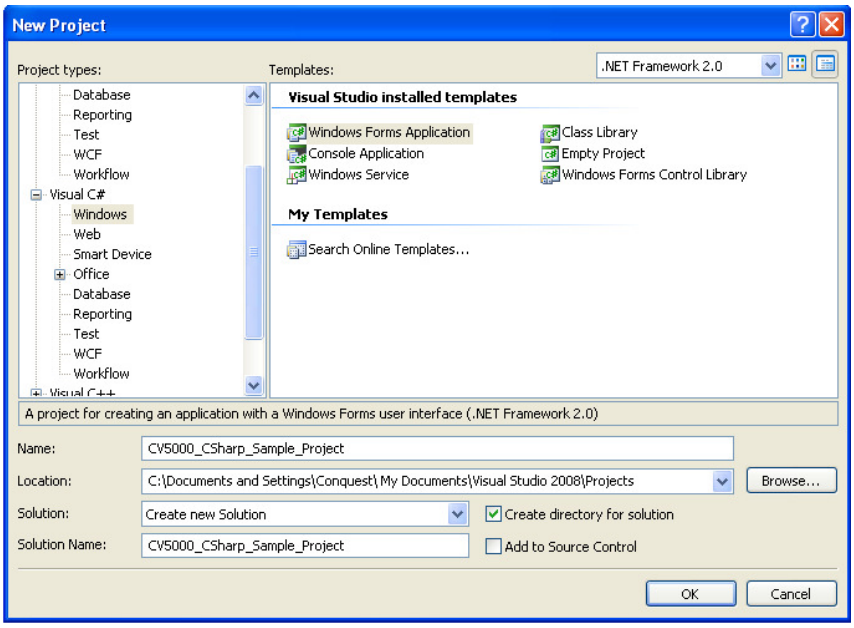
Configuring the CV-5000 is beyond the scope of this document and is not discussed here. Please consult your company's IT department and refer to the CV-5000 User's Manual for more information.

The sample program will be responsible for triggering the CV-5000 every 5 seconds. When measurement and image data is received by the CV-5000 via the Ethernet connection it will be displayed to the user and saved to disk. This program will use a single camera and a single window.

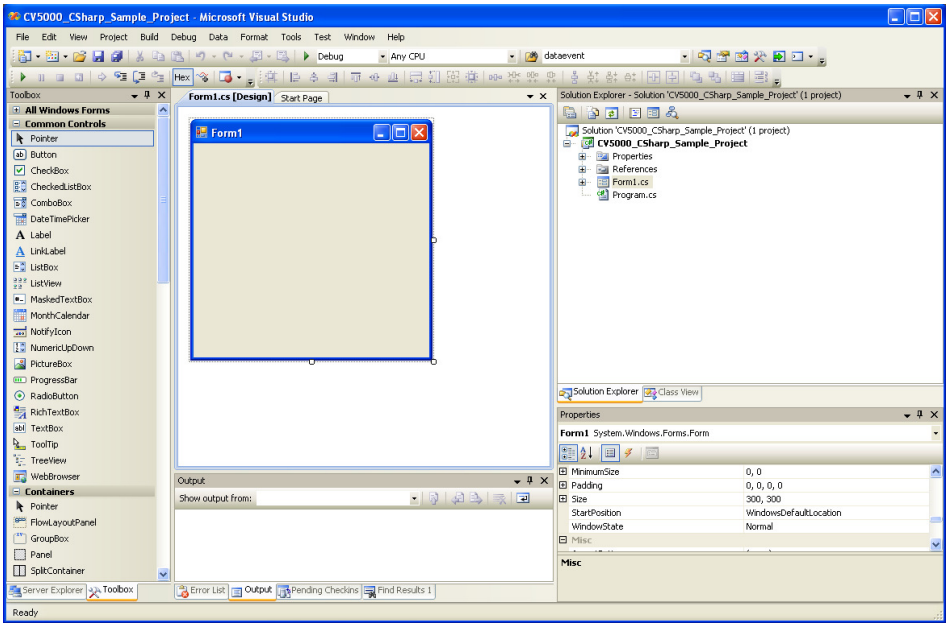
Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.1 Setting up your Program

Launch Visual Studio 2008 and create a new C# Forms application by selecting a Visual C# project type along with the Windows Forms Application project template (NOTE: The SDK can be used with any .NET language such as VB.Net). Make sure that the .NET Framework is set to 2.0 and give the project the name CV5200_Sample_Project as shown below. Then press the OK button to continue.



Once you click on OK you should have a blank project open that looks similar to the screen below. If the solution explorer is not open (see right hand side of image below), you will need to open it by clicking ***View … Solution Explorer*** on the Visual Studio menu.

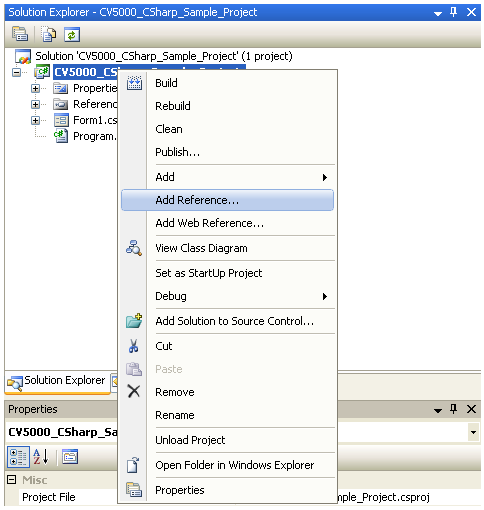


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

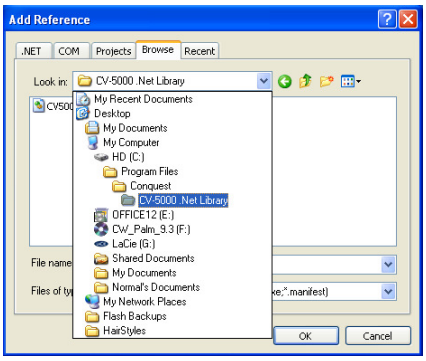
Adding a Reference to the CV-5000 .NET Library

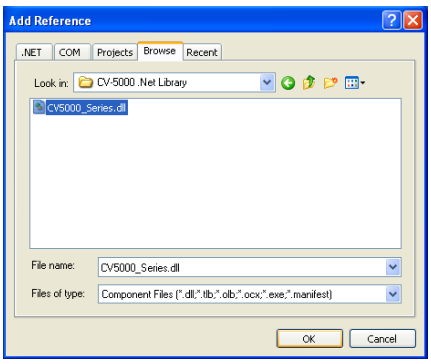
Before you can use the CV-5000 .NET Library in your application you must add a reference to it in your code. This will allow you to access the properties and methods of the CV-5000 .NET Library.

1. In the Solution Explorer, right click on CV5000_Sample_Project and select Add Reference…



2. Click on the browse tab and navigate to the directory where the CV-5000 .NET Library was installed. The default directory is C:\Program Files\Conquest\CV-5000 .Net Library. Select CV5000_Series.dll and then click OK.



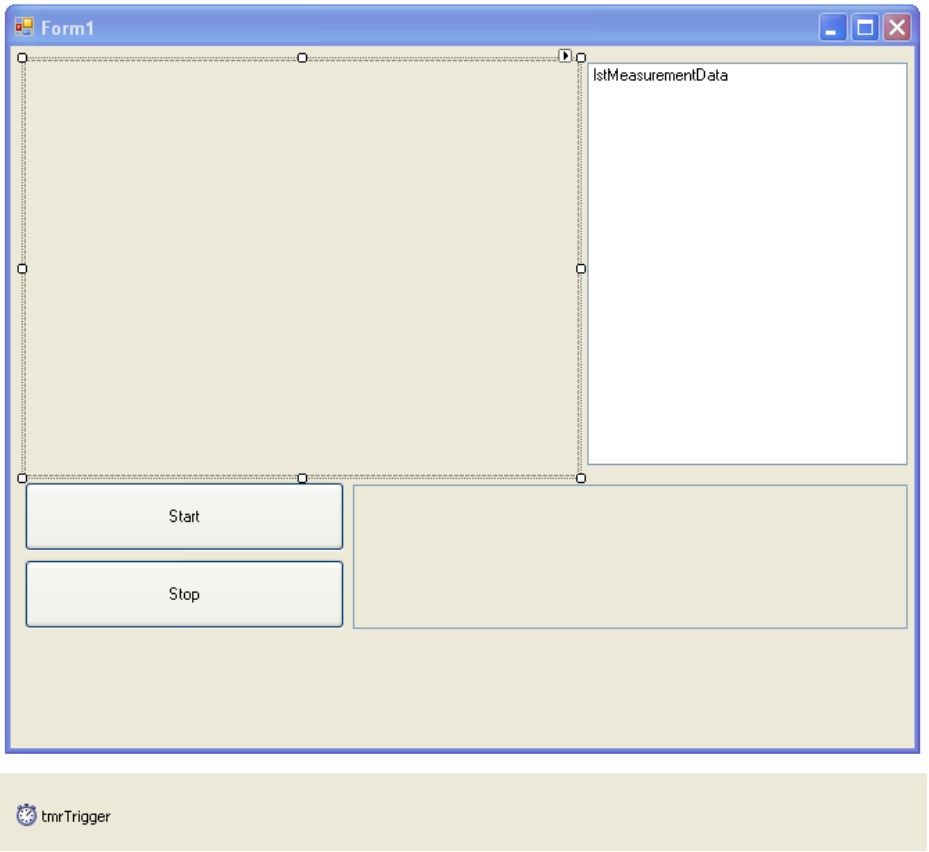


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.2 Add Required GUI Components to the Application's Main Form

1. Add a PictureBox to the form and rename it to picPartImage. Set the SizeMode to StretchImage and place in the top left of the form.
2. Add a ListBox to the form and rename it lstMeasurementData. Place in the top right of the form.
3. Add a Button to the form and rename it to btnStart. Set the Text Property to "Start".
4. Add another button to the form and rename it to btnStop. Set the Text Property to "Stop".
5. Add Timer control to the form and rename it to tmrTrigger.
6. Add a TextBox to the form and rename it to txtStatus. Set the ReadOnly property and the Multiline property to true. Place txtStatus in the bottom right of the form.

After all the controls are laid out, the form should look similar the following image:



Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.
Conquest Consulting LLC

1.3 Connecting to the CV-5000 Device

In this section we will be adding code to the application, the light gray code is code that has already been added to the application either by the windows IDE or in a previous step and is just included to help the user grasp where the code is to be added. In this section we will add the code to connect to the CV-5000 device and display the connection status on the screen so the user can see that they are connected. This section will also introduce the basic concepts of thread-safe programming and handling the events that are raised by the CV-5000 Library. The sections that follow will build on these basic concepts to retrieve measurement data and finally an image from the CV-5000.

1.3.1 Add CV5000_Series Namespace & Create an CV-5000 Library Object

Use the *using* statement to enable access to the CV-5000 library's namespace. Before you add the *using* statement you must have previously added a reference to the library as defined in Section 5.2. Now double click on the title of the form to bring up the forms code window and add the following above the forms class definition as shown:

```
using CV5000_Series;

namespace CV5000_CSharp_Sample_Project
{
        public partial class Form1 : Form
        {

                public Form1()
                {
                        InitializeComponent();
                }

        }
}
```

For this sample program we are going to create a CV-5000 object that is global to Form1. Add the following code just below the forms class definition as shown below:

```
using CV5000_Series;

namespace CV5000_CSharp_Sample_Project
{
        public partial class Form1 : Form
        {
                CV5000 cv5000;

                public Form1()
                {
                        InitializeComponent();
                }

        }
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.3.2 Connecting to the CV

The sample program will connect to the CV-5000 as soon as it starts up so we will need to create an instance of the CV-5000 library, enter the password, and configure it to connect to the CV-5000 when the form loads. This is accomplished by adding the following code to the form's load event as shown below:

```
private void Form1_Load(object sender, EventArgs e)
{
     //Create an instance of the CV-5000 library
     cv5000 = new CV5000();

     //Set up the CV-5000 I.P. address and port information
     cv5000.RemotePort = 8500;
     cv5000.RemoteImagePort = 8501;
     cv5000.RemoteHost = "192.168.168.211";

     //Connect to the CV-5000 series device
       cv5000.Connect();
}
```

NOTE - For the RemoteHost property, use the IP address that was configured in the CV-5000 series device being connected to.

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

1.3.3 Creating Event Handlers

In the last section you created a connection to the CV-5000 but we do not know whether or not the connection was successful. This is because the CV-5000 .NET Library communicates with the application through events. A complete list of the events is contained in Section 6.2. The sample application will use a subset of these events. It is through these events that your application will receive the data and any status information that the CV-5000 returns.

To set up your code to be able to handle these events you first have to create "hooks" into the CV-5000 .NET Library's events and then you have to create the event handlers in your application. Since we want to be able to get feedback from the Connect method we must add these hooks before we connect to the CV-5000. Add the following code to the form's load method as shown below.

```
cv5000.RemoteHost = "192.168.168.211";

   cv5000.ConnectionEstablished += new CV5000.ConnectionEstablishedHandler(
       cv5000_ConnectionEstablished);
   cv5000.ImageConnectionEstablished += new CV5000.ImageConnectionEstablishedHandler(
       cv5000_ImageConnectionEstablished);
   cv5000.ErrorOccurred += new CV5000.ErrorOccurredHandler(cv5000_ErrorOccurred);

   //Connect to the CV-5000 series device
  cv5000.Connect();
```

Now that the hooks are in place we can add the event handlers. Add the following methods to the form's class definition. The `SetStatus` method is defined in the next section.

```
void cv5000_ConnectionEstablished()
{
        //Triggered when the command port is connected to
        SetStatus("Command port connected");
}

void cv5000_ImageConnectionEstablished()
{
        //Triggered when the image port is connected to
        SetStatus("Image port connected");
}

void cv5000_ErrorOccurred(ErrorOccurredEventArgs args)
{
        if (args.ExceptionCaught != null)
        {
                MessageBox.Show(args.ErrorMessage + "," +
                args.ExceptionCaught.ToString());
        }
        else    MessageBox.Show(args.ErrorMessage);
}
```

1.3.4 Updating the Status Textbox in a Thread-safe Manner

The CV-5000 .NET Library uses independent threads to poll for data from the CV-5000 device. Because of this, it is necessary to create your application in a thread-safe manner. Since Windows forms controls are not thread-safe the controls on the form cannot be directly updated from the event handlers in the application. Therefore the creation of a thread-safe method is required to update any of the controls on the form.

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

The sample application has a status field that is updated whenever a significant event has happened. The code below shows how to use the CV-5000 .NET Library in a thread-safe way to update the status field in the application. Add the following methods to the form's class.

```
//Function definition used to update GUI in a threadsafe manner
delegate void UpdateStringDelegate(string entry);

//Called with an invoke when udating txtStatus in event handler
private void UpdateStatus(string entry)
{
        txtStatus.Text = entry + "\r\n" + txtStatus.Text;
}

//Update status TextBox using invoke
private void SetStatus(string entry)
{
        Invoke(new UpdateStringDelegate(UpdateStatus), entry);
}
```

The `SetStatus` method uses the delegate method `UpdateStringDelegate` to call the `UpdateStatus` method. This is necessary because Windows Forms controls are not thread safe.

As you can see from the code above the `UpdateStringDelegate` defines a method signature that can be used with the Invoke Method to update a Windows Forms GUI control and its signature is compatible with the `UpdateStatus` method. To boil the Microsoft tech-speak into English what this means is that the `UpdateStringDelegate` method and the `UpdateStatus` method have the same return type (in this case none) and the same parameter list.

NOTE - See the following article in Microsoft's documentation for more information on this concept. http://msdn.microsoft.com/en-us/library/system.windows.forms.control.invokerequired.aspx

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

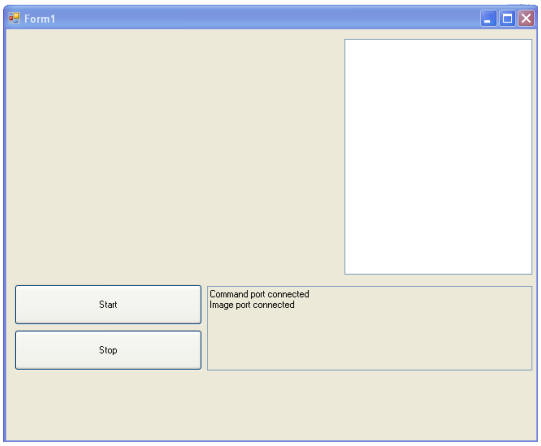
1.3.5 Closing Connections

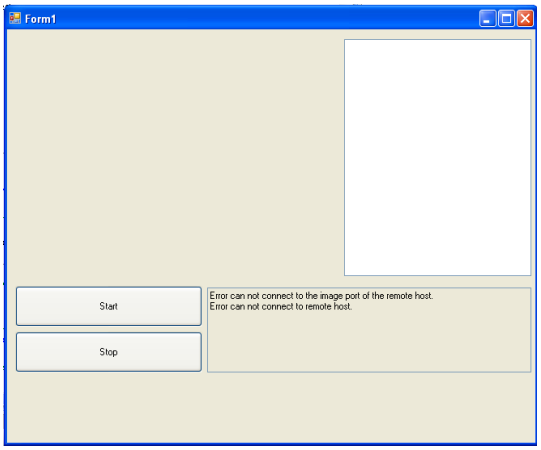
Before the application shuts down, you should close your connection to the CV-5000. To do this the CloseConnection() method in the CV-5000 .NET Library should be called. Add a form closing method to the sample application and add a call to CloseConnection() as shown below.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
        cv5000.CloseConnection();
}
```

1.3.6 Running the Application with Connection Events

In its present state, all the application does is connect to the CV-5000 series device. Run the application. Since the connect command is in the Form1_Load event handler, once you run the program you should see something similar to the image on the left below. (It may take a second or two for the "command port connected" text to appear) If the application is not able to connect to the CV-5000 series device, the error message(s) shown in the image on the right will appear after several seconds.





If you ran into an error while creating the code in this section, Section 2 contains a listing of the code up to this point.

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.4 Retrieving Measurement Data from the CV-5000 Device

In this section we will be expanding the code that we created in section 5.4 to retrieve measurement data from the CV-5000 and display it in the measurement list box. We will introduce a new event (DataReceived) and set up the Start and Stop buttons.

1.4.1 Updating the Measurement ListBox in a ThreadSafe way

The sample application has a measurement listbox that is updated each time measurement data is received. The code below shows how to use the CV-5000 .NET Library in a thread-safe way to update the measurement listbox in the application. Add the following methods to the form class.

```
//Add data received to measurement list box
private void UpdateMeasurementData(string entry)
{
lstMeasurementData.Items.Add(entry);
}

//Update Measurement list box, use invoke if necessary
private void SetMeasurementData(string entry)
{
Invoke(new UpdateStringDelegate(UpdateMeasurementData), entry);
}
```

The `SetMeasurementData` method invokes `UpdateMeasurementData` method because the `SetMeasurementData` method will be used in an event triggered from a separate thread.

NOTE - The `UpdateStringDelegate` method is the same method that was created in section 5.4.4 so it is not recreated here. You will notice that the `UpdateMeasurementData` method has a compatible signature which is why we can use the same delegate.

NOTE - See the following article in Microsoft's documentation for more information on this concept. http://msdn.microsoft.com/en-us/library/system.windows.forms.control.invokerequired.aspx

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.4.2 Add Data Received Event Handler

When the CV-5000 .NET Library receives data from the CV-5000 it raises the Data Received event. To retrieve the data you must "hook" into the event and add a handler for it. First hook into the event in the form's load method with the rest of the handlers as shown below:

```
cv5000.ConnectionEstablished += new CV5000.ConnectionEstablishedHandler(
    cv5000_ConnectionEstablished);
cv5000.ImageConnectionEstablished += new CV5000.ImageConnectionEstablishedHandler(
    cv5000_ImageConnectionEstablished);
cv5000.ErrorOccurred += new CV5000.ErrorOccurredHandler(cv5000_ErrorOccurred);
cv5000.DataReceived += new CV5000.DataReceivedHandler(cv5000_DataReceived);

//Connect to the CV-5000 series device
cv5000.Connect();
```

Now that the hooks are in place we can add the event handler to our application. Add the following method to the form class.

```
void cv5000_DataReceived(DataEventArgs args)
{
        string str;

        //The data from the CV-5000 is stored in 'args'
        if (args.ASCIIData.Length > 30)
                str = args.ASCIIData.Substring(0, 30) + "...";
        else
                str = args.ASCIIData;

        if (args.RecMode == CVRecMode.CVError)
                SetMeasurementData(str);
        else if (args.Port == CVPort.CVDataPort)
                SetMeasurementData(str);
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.4.3 Setting up the Timer

The timer will be the source of triggers to the CV-5000 series device. Each time the timer expires it will send a trigger to the CV. Double click on the tmrTrigger control to add a handler for the timer event and add the following code to the event.

```
private void tmrTrigger_Tick(object sender, EventArgs e)
{
        cv5000.Trigger(1);
}
```

1.4.4 Setting up the Start and Stop Buttons

The start and stop buttons on the form will enable and disable the timer on the form. Create event handlers by double clicking on the buttons and add the following code to the application.

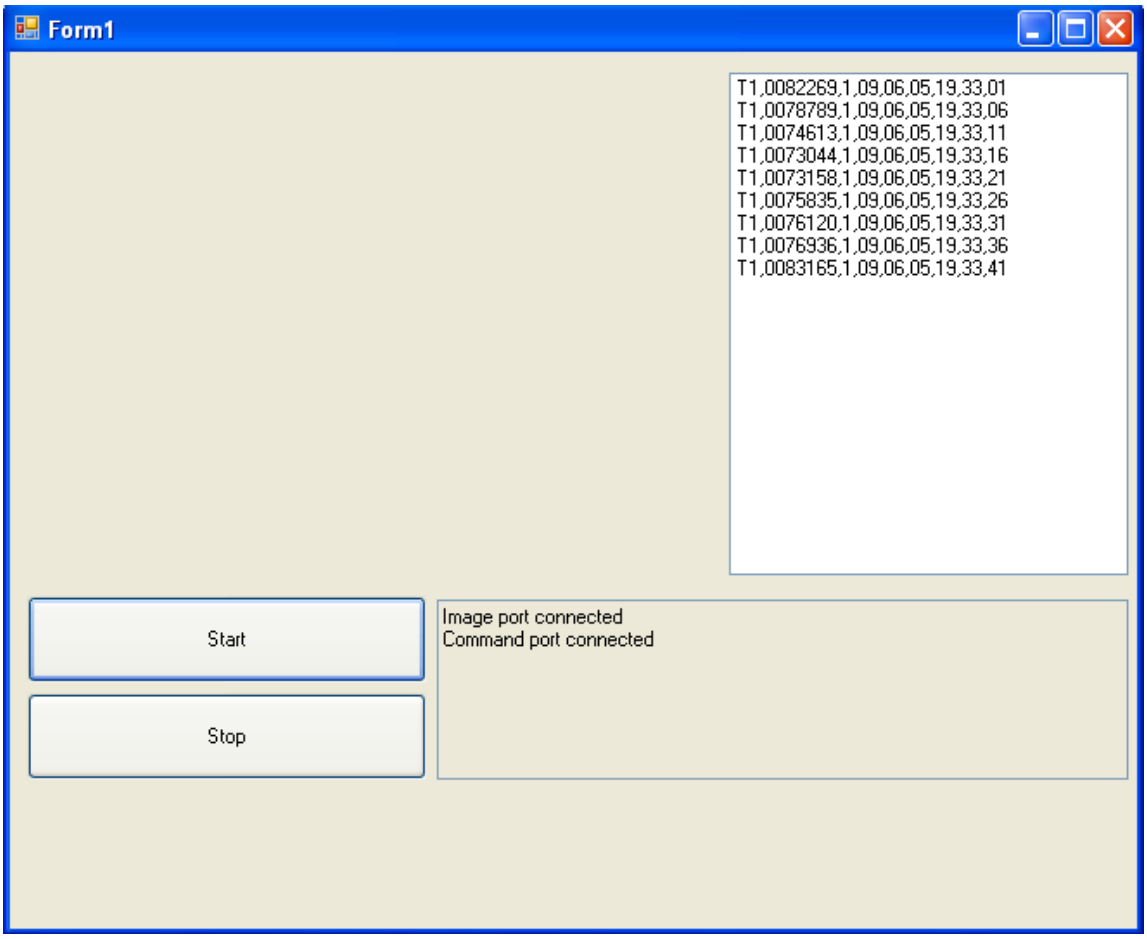
```
private void btnStart_Click(object sender, EventArgs e)
{
        //Trigger the CV-5000 device now
        cv5000.Trigger(1);

        tmrTrigger.Interval = 5000;
        tmrTrigger.Enabled = true;
}

private void btnStop_Click(object sender, EventArgs e)
{
        tmrTrigger.Enabled = false;
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

1.4.5 Running the Application with the Trigger Data

Run the application and press the start button. The output should look similar to the following screen shot on the right. The actual data shown will depend on the configuration of the program in the CV-5000 series device. The data will be received once every 5 seconds so you may want to wait for about 20 seconds or so to elapse.



If you ran into an error while creating the code in this section, Section 3 contains a listing of the code up to this point.

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.5 Retrieving Image Data from the CV-5000Device

In this section we will be expanding the code that we created in section 5.5 to retrieve image data from the CV-5000 and display it in the picture box.

1.5.1 Updating the PictureBox in a Thread-safe Manner

The sample application has a picture box that is updated each time image data is received. The code below shows how to use the CV-5000 .NET Library in a thread-safe way to update the picture box in the application. Add the following methods to the form class.

```
delegate void UpdateBitmapDelegate(Bitmap bmp);

//Add image to picturebox
private void UpdateBitmap(Bitmap bmp)
{
        Bitmap oldBitmap;

        oldBitmap = (Bitmap) picPartImage.Image;
        if (oldBitmap != null)
        {
                //Release windows resources
                oldBitmap.Dispose();
        }

        picPartImage.Image = bmp;
}

//Update picturebox using invoke
private void setBitmap(Bitmap bmp)
{
        Invoke(new UpdateBitmapDelegate(UpdateBitmap), bmp);
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.5.2 Add Image Data to the Data Received Handler

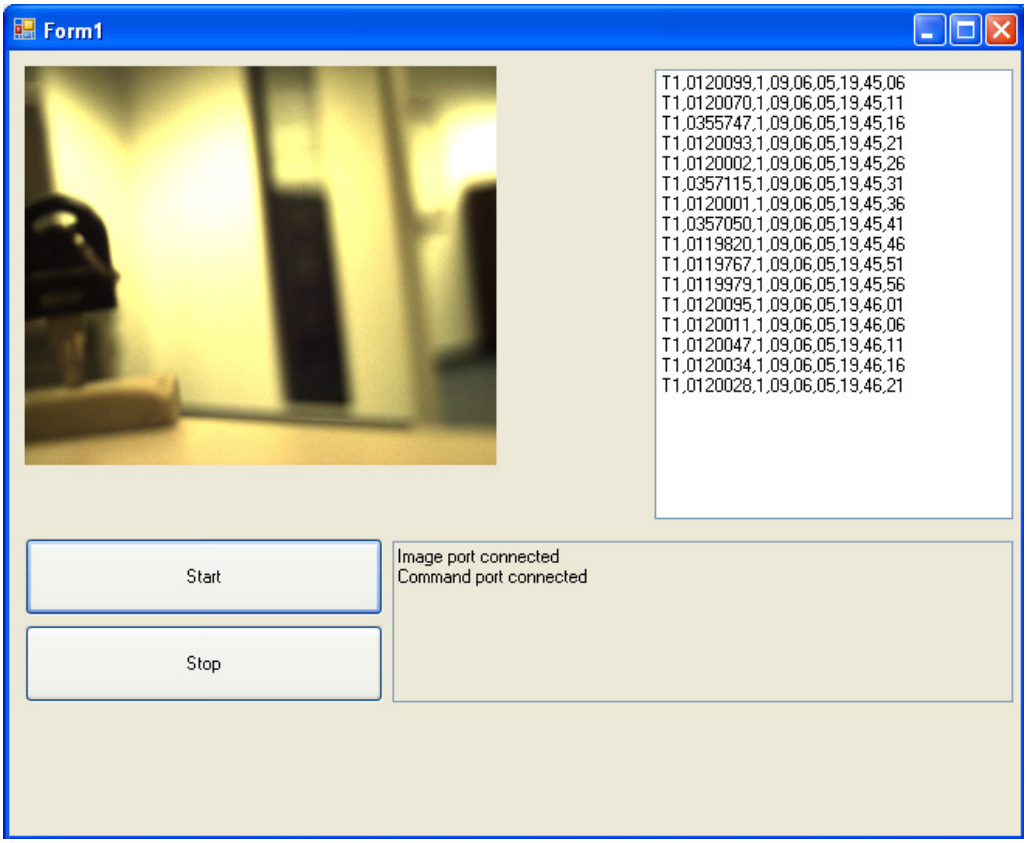
When the data is received we need to call the correct handler based on the type of data being received, so add the following code to the end of the DataReceived event handler.

```
void cv5000_DataReceived(DataEventArgs args)
{
        string str;

        //The data from the CV-5000 is stored in 'args'
        if (args.ASCIIData.Length > 30)
                str = args.ASCIIData.Substring(0, 30) + "...";
        else
                str = args.ASCIIData;

        if (args.RecMode == CVRecMode.CVError)
                SetMeasurementData(str);
        else if (args.Port == CVPort.CVDataPort)
                SetMeasurementData(str);

        if (args.RecMode == CVRecMode.TrigImageData || args.RecMode ==
            CVRecMode.RegImageBMP || args.RecMode == CVRecMode.ScreenCapture)
        {
                setBitmap(args.Bitmap);
        }
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

1.5.3 Running the Complete Sample Program

Run the program and the program should display both the measurement data and an image with each trigger. You may want to pass your hand in front of the camera to be sure that the image is changing. Again the image will be triggered once every 5 seconds so you may want to wait a few seconds.



If you ran into an error while creating the code in this section, Appendix A.3 contains a listing of the complete code.

CONGRATULATIONS!!! You have just completed your first program.

Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

2 CONNECTION CODE LISTING

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using CV5000_Series;

namespace CV5000_CSharp_Sample_Project
{
    public partial class Form1 : Form
    {
        CV5000 cv5000;

        //Delegate defines function used to update GUI in a threadsafe manner
        delegate void UpdateStringDelegate(string entry);

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Create an instance of the CV-5000 library
            cv5000 = new CV5000();

              //Set up the CV-5000 I.P. address and port information
            cv5000.RemotePort = 8500;
            cv5000.RemoteImagePort = 8501;
            cv5000.RemoteHost = "192.168.168.211";

               cv5000.ConnectionEstablished += new CV5000.ConnectionEstablishedHandler(
                   cv5000_ConnectionEstablished);
               cv5000.ImageConnectionEstablished += new CV5000.ImageConnectionEstablishedHandler(
                   cv5000_ImageConnectionEstablished);
               cv5000.ErrorOccurred += new CV5000.ErrorOccurredHandler( cv5000_ErrorOccurred);

            //Connect to the CV-5000 series device
            cv5000.Connect();

        }

        //Called with an invoke when udating txtStatus in event handler
        private void UpdateStatus(string entry)
        {
            txtStatus.Text = entry + "\r\n" + txtStatus.Text;
        }

        //Update status TextBox using invoke
        private void SetStatus(string entry)
        {
            Invoke(new UpdateStringDelegate(UpdateStatus), entry);
        }

        void cv5000_ConnectionEstablished()
        {
            //Triggered when the command port is connected to
            SetStatus("Command port connected");
        }

        void cv5000_ImageConnectionEstablished()
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

```
        {
            //Triggered when the image port is connected to
            SetStatus("Image port connected");
        }

        void cv5000_ErrorOccurred(ErrorOccurredEventArgs args)
        {
            if (args.ExceptionCaught != null)
            {
                      MessageBox.Show(args.ErrorMessage + "," +
                          args.ExceptionCaught.ToString());
            }
            else MessageBox.Show(args.ErrorMessage);

        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            cv5000.CloseConnection();
        }
    }
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

3 MEASUREMENT DATA CODE LISTING

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using CV5000_Series;

namespace CV5000_CSharp_Sample_Project
{
    public partial class Form1 : Form
    {
        CV5000 cv5000;

        //Delegate defines function used to update GUI in a threadsafe manner
        delegate void UpdateStringDelegate(string entry);

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Create an instance of the CV-5000 library
            cv5000 = new CV5000();

            //Set up the CV-5000 I.P. address and port information
            cv5000.RemotePort = 8500;
            cv5000.RemoteImagePort = 8501;
            cv5000.RemoteHost = "192.168.168.211";

               cv5000.ConnectionEstablished += new CV5000.ConnectionEstablishedHandler(
                   cv5000_ConnectionEstablished);
               cv5000.ImageConnectionEstablished += new CV5000.ImageConnectionEstablishedHandler(
                   cv5000_ImageConnectionEstablished);
               cv5000.ErrorOccurred += new CV5000.ErrorOccurredHandler( cv5000_ErrorOccurred);
               cv5000.DataReceived += new CV5000.DataReceivedHandler(cv5000_DataReceived);

            //Connect to the CV-5000 series device
            cv5000.Connect();
        }

        //Called with an invoke when udating txtStatus in event handler
        private void UpdateStatus(string entry)
        {
            txtStatus.Text = entry + "\r\n" + txtStatus.Text;
        }

        //Update status TextBox using invoke
        private void SetStatus(string entry)
        {
            Invoke(new UpdateStringDelegate(UpdateStatus), entry);
        }

        void cv5000_DataReceived(DataEventArgs args)
        {
            string str;

            //The data from the CV-5000 is stored in 'args'
            if (args.ASCIIData.Length > 30)
                str = args.ASCIIData.Substring(0, 30) + "...";
            else
                str = args.ASCIIData;
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

```
            if (args.RecMode == CVRecMode.CVError)
                SetMeasurementData(str);
            else if (args.Port == CVPort.CVDataPort)
                SetMeasurementData(str);
        }

        //Add data received to measurement list box
        private void UpdateMeasurementData(string entry)
        {
            lstMeasurementData.Items.Add(entry);
        }

        //Update Measurement list box, use invoke if necessary
        private void SetMeasurementData(string entry)
        {
            Invoke(new UpdateStringDelegate(UpdateMeasurementData), entry);
        }

        void cv5000_ConnectionEstablished()
        {
            //Triggered when the command port is connected to
            SetStatus("Command port connected");
        }

        void cv5000_ImageConnectionEstablished()
        {
            //Triggered when the image port is connected to
            SetStatus("Image port connected");
        }

        void cv5000_ErrorOccurred(ErrorOccurredEventArgs args)
        {
            if (args.ExceptionCaught != null)
            {
                      MessageBox.Show(args.ErrorMessage + "," +
                          args.ExceptionCaught.ToString());
            }
            else MessageBox.Show(args.ErrorMessage);
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            cv5000.CloseConnection();
        }

        private void tmrTrigger_Tick(object sender, EventArgs e)
        {
            cv5000.Trigger(1);
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            //Trigger the CV-5000 device now
            cv5000.Trigger(1);

            tmrTrigger.Interval = 5000;
            tmrTrigger.Enabled = true;
        }

        private void btnStop_Click(object sender, EventArgs e)
        {
            tmrTrigger.Enabled = false;
        }
    }
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

4 COMPLETE CODE LISTING

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using CV5000_Series;

namespace CV5000_CSharp_Sample_Project
{
    public partial class Form1 : Form
    {
        CV5000 cv5000;

        //Delegates defines function used to update GUI - threadsafe manner
        delegate void UpdateStringDelegate(string entry);
        delegate void UpdateBitmapDelegate(Bitmap bmp);

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Create an instance of the CV-5000 library
            cv5000 = new CV5000();

            //Set up the CV-5000 I.P. address and port information
            cv5000.RemotePort = 8500;
            cv5000.RemoteImagePort = 8501;
            cv5000.RemoteHost = "192.168.168.211";

               cv5000.ConnectionEstablished += new CV5000.ConnectionEstablishedHandler(
                   cv5000_ConnectionEstablished);
               cv5000.ImageConnectionEstablished += new CV5000.ImageConnectionEstablishedHandler(
                   cv5000_ImageConnectionEstablished);
               cv5000.ErrorOccurred += new CV5000.ErrorOccurredHandler( cv5000_ErrorOccurred);
               cv5000.DataReceived += new CV5000.DataReceivedHandler(cv5000_DataReceived);

            //Connect to the CV-5000 series device
            cv5000.Connect();
        }

        //Called with an invoke when udating txtStatus in event handler
        private void UpdateStatus(string entry)
        {
            txtStatus.Text = entry + "\r\n" + txtStatus.Text;
        }

        //Update status TextBox using invoke
        private void SetStatus(string entry)
        {
            Invoke(new UpdateStringDelegate(UpdateStatus), entry);
        }

        void cv5000_DataReceived(DataEventArgs args)
        {
            string str;

            //The data from the CV-5000 is stored in 'args'
            if (args.ASCIIData.Length > 30)
                str = args.ASCIIData.Substring(0, 30) + "...";
            else
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

```
            str = args.ASCIIData;

        if (args.RecMode == CVRecMode.CVError)
            SetMeasurementData(str);
        else if (args.Port == CVPort.CVDataPort)
            SetMeasurementData(str);

           if (args.RecMode == CVRecMode.TrigImageData || args.RecMode ==
               CVRecMode.RegImageBMP || args.RecMode == CVRecMode.ScreenCapture)
        {
            setBitmap(args.Bitmap);
        }
    }

    //Add data received to measurement list box
    private void UpdateMeasurementData(string entry)
    {
        lstMeasurementData.Items.Add(entry);
    }

    //Update Measurement list box, use invoke if necessary
    private void SetMeasurementData(string entry)
    {
        Invoke(new UpdateStringDelegate(UpdateMeasurementData), entry);
    }

    //Add image to picturebox
    private void UpdateBitmap(Bitmap bmp)
    {
        Bitmap oldBitmap;

        oldBitmap = (Bitmap) picPartImage.Image;
        if (oldBitmap != null)
        {
            //Release windows resources
            oldBitmap.Dispose();
        }

        picPartImage.Image = bmp;
    }

    //Update picturebox using invoke
    private void setBitmap(Bitmap bmp)
    {
        Invoke(new UpdateBitmapDelegate(UpdateBitmap), bmp);
    }

    void cv5000_ConnectionEstablished()
    {
        //Triggered when the command port is connected to
        SetStatus("Command port connected");
    }

    void cv5000_ImageConnectionEstablished()
    {
        //Triggered when the image port is connected to
        SetStatus("Image port connected");
    }

    void cv5000_ErrorOccurred(ErrorOccurredEventArgs args)
    {
        if (args.ExceptionCaught != null)
        {
                  MessageBox.Show(args.ErrorMessage + "," +
                      args.ExceptionCaught.ToString());
        }
        else MessageBox.Show(args.ErrorMessage);

    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC

```
        {
            cv5000.CloseConnection();
        }

        private void tmrTrigger_Tick(object sender, EventArgs e)
        {
            cv5000.Trigger(1);
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            //Trigger the CV-5000 device now
            cv5000.Trigger(1);

            tmrTrigger.Interval = 5000;
            tmrTrigger.Enabled = true;
        }

        private void btnStop_Click(object sender, EventArgs e)
        {
            tmrTrigger.Enabled = false;
        }
    }
}
```


Use or disclosure of data on this page is subject to the restrictions in the End-User License Agreement of this document.

Conquest Consulting LLC